國立臺北科技大學

資訊工程系碩士班

碩士學位論文

以起音點偵測為基礎的歌曲推薦系統之效能評估

Performance Evaluation of Music Recommendation System Based on Onset Detection

研究生: 施賀翔

指導教授:尤信程

中華民國一百零七年七月

摘要

論文名稱：以起音點偵測為基礎的歌曲推薦系統之效能評估

頁數：三十六頁

校所別：國立台北科技大學資訊工程系研究所

畢業時間：一百零六學年度 第二學期

學位：碩士

研究生：施賀翔

指導教授：尤信程 博士

關鍵詞： 起音點偵測、歌曲推薦系統、效能評估

本論文承接趙若偉學長所做的「利用起音點偵測搭配 RLCS 演算法進行歌曲相似度比對」，將其中所使用的起音點偵測方法與專門處裡音訊相關的套件 librosa 與 madmom 做比較，評估起音點偵測準確度差異及其對相似度比較的影響。

並且與歌曲相似度比對函式庫 musly 進行聆聽實驗比較，評估不同系統所推薦之歌曲實際上人耳所感受到相似度差異。

另外本次系統實作從原本使用的 matlab 轉移到 python 上，在改寫的同時利用 Cython 以及平行化處理等方式大幅度改善極為耗時的 RLCS 演算法的計算效率。

ABSTRACT

Title：Performance Evaluation of Music Recommendation System
Based on Onset Detection
Page：36
School：National Taipei University of Technology
Department：Computer Science and Information Engineering
Time：July, 2018
Degree：Master
Researcher：Ho-Hsiang Shih
Advisor：Shing-Chern D.You, Ph.D.

Keyword：Onset Detection, Music Recommendation,Performance Evaluation

This thesis is follow up of the thesis “Music Matching using Onset Detection with RLCS Algorithm” write by the upperclassman Ro-Wei Chao. We compare the onset detection method with the music information retrieval package librosa and madmom. Evaluate the performance of onset detection and how it affect the music similarity comparison.

And we do an experiment of music listening between our system and a music similarity library musly. In order to know how human’s ear feel the music similarity between these two systems.

In addition, the program implement is transfer from matlab to python. And we use Cython and parallel computing to improve performance of RLCS algorithm witch is slowly and waste a lot of time.

目錄

表目錄

圖目錄

第一章　緒論

1.1 前言

在現代社會，各種的娛樂媒體，不論是音樂、影集、遊戲或者書籍，人們接收的媒介或快或慢，都由傳統的實體物(卡匣、CD、紙本)，逐漸地轉向藉由網路傳播的數位化方式，當使用者接觸這些提供數位內容的服務時，系統如何讓使用者能更容易地找到所期望的內容，就成為了重要的課題。

而在音樂媒體方面，常見的方式是由使用者輸入，歌手、年代、歌詞等關鍵字來搜尋，此種方式是以文字資訊為基礎，仰賴使用者對歌曲的知識。但是，當使用者只有片段的音樂旋律，不知其詳細資訊，例如在店面中偶然聽到的歌曲，或是插入在某個影片片段中的背景音樂；或是想要找尋與某首歌曲相似的歌曲時，就無法以文字的方式來描述。這時候就需要以內容為基礎的方式，讓系統從歌曲片段中分析有用的資訊，來找尋使用者可能想要找的歌曲。

1.2 研究動機與目的

本研究承接趙若偉學長所做的「利用起音點偵測搭配 RLCS 演算法進行歌曲相似度比對」[1]，該研究旨在以起音點偵測(onset detection)為基礎的方式，提供使用者相似的歌曲。

所謂的起音點，也就是聲音訊號開始的點，在起音點所在位置會有明顯的能量變化。而在歌曲中，起音點位置的間隔時間，可以用來描述歌曲在節奏方面的特徵。

在本研究中，首先將前研究所使用的起音點偵測方式，與專門處理音訊相關的套件 librosa[2]與 madmom[3]做比較，評估不同系統間起音點偵測準確度的差異，以及起音點偵測準確度對歌曲相似度比較結果的影響。

接下來將歌曲相似度比較的結果，與歌曲相似度比對函式庫 musly[4]進行比較，以聆聽實驗的方式評估不同系統所推薦的歌曲，實際上人耳所感受到相似度差異。並且將其中在計算上極為耗時的概略最長共同子序列 (Rough Longest Common Subsequence, RLCS)演算法，在將系統實作從 matlab 轉移到

python 的同時，以 Cython 以及平行化處理等方式，來改善其計算效率。

最後，進行了一小部份的延伸研究，對於不同的樂曲曲風，如華爾滋、探戈、森巴等，嘗試以起音點偵測為基礎的方式，找出樂曲的節奏，並進行曲風的判斷以及分類。

綜合上述，期望與其他系統的比較與實驗來評估前研究所使用方法的效能，並在之上做進一步的改進與延伸。

1.3 章節編排

本論文共分為五章，第一章為緒論並簡述研究目的與動機，第二章則為介紹各系統所使用的起音點偵測方式並且進行比較，第三章則是不同歌曲推薦系統的介紹與聆聽實驗結果，第四章則說明在系統實作上所做的計算效率改善方式與成果，第五章為延伸研究與未來展望。

第二章　起音點偵測

本章節將分別介紹各系統所使用的起音點偵測方式，第一節將針對前研究所使用的起音點偵測的流程作簡單的介紹，第二節則分別介紹兩個比較對象系統所使用的起音點偵測方式，第三節則是比較各系統的起音點偵測準確率與歌曲相似度比對結果。

一般的起音點偵測流程如圖 2.1 所示，包含了前處理、維度降低、峰值擷取以及決策四個步驟。

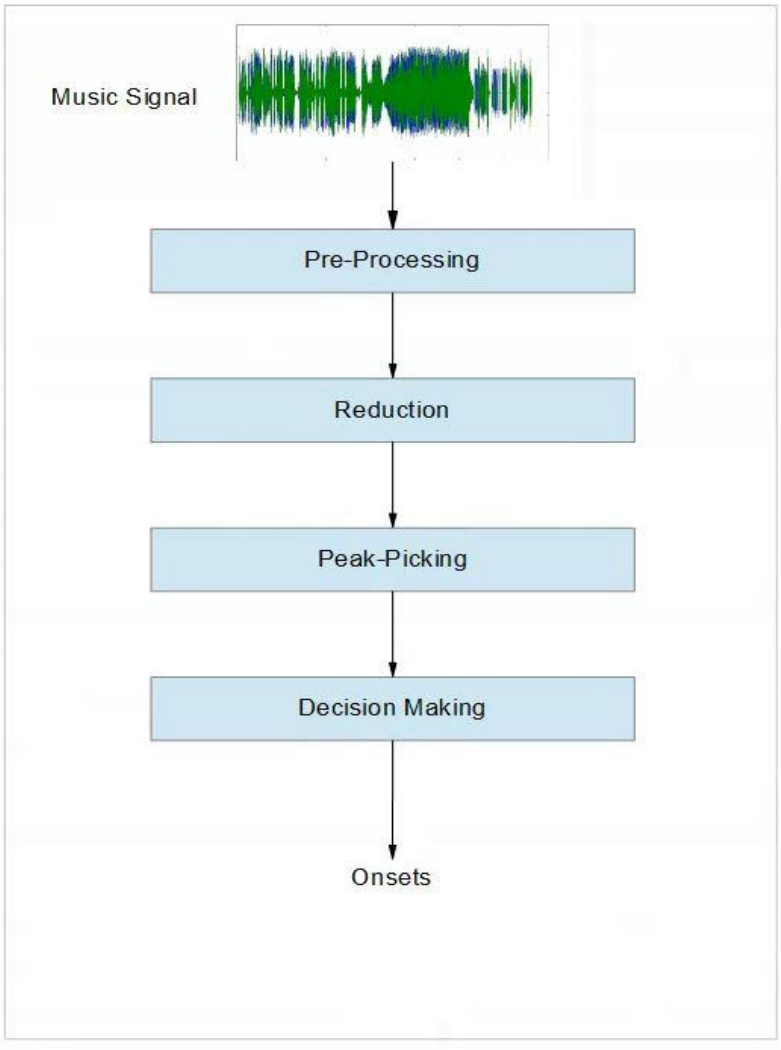

圖 2.1 起音點偵測流程，取材自[1]

在讀取原始的音訊資料後，首先會經過前處理(Pre-Processing)，將訊號濾波或做時頻轉換，以利之後的分析。接著是降低維度(Reduction)階段，經過前

處理的訊號，會再經過偵測函數(Detection Function)處理，轉換為較低維度的特徵數列。第三步驟為峰值擷取(Peak-Picking)，從經過前兩步驟處理的數列中，找出可能為起音點的位置。在最後的決策(Decision Making)階段，將被選出的可能為起音點的位置進一步篩選，得到最後的結果。

其中前兩步驟的前處理與降低維度所使用的方法，是決定起音點偵測準確性的關鍵階段，因此在之後起音點偵測方法的比較上，也會集中於這兩個步驟。

2.1前研究

前研究為趙若偉學長所做的「利用起音點偵測搭配 RLCS 演算法進行歌曲相似度比對」[1]，在建構偵測函數的部分，同時用了時間特徵(Temporal Feature)與頻譜特徵(Spectrum Feature)的方法。

在時間特徵偵測函數的部分，首先通過濾波器將音樂訊號切割為四個不重疊(Non-Overlapping)頻段的訊號，分別為 630hz 以下、630hz~1720hz、720hz~4400hz 以及 4400hz 以上，這些取樣點再以 512 個取樣點重疊(Overlap)一半的方式分為訊框(frame)，乘上漢明窗(hamming window)後，計算每個訊框的總能量，成為一個特徵點，如公式(2.1)。

$$\mathrm{E(n)} = \frac{1}{N}\sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} |x(n+m)|^2 w(m) \tag{2.1}$$

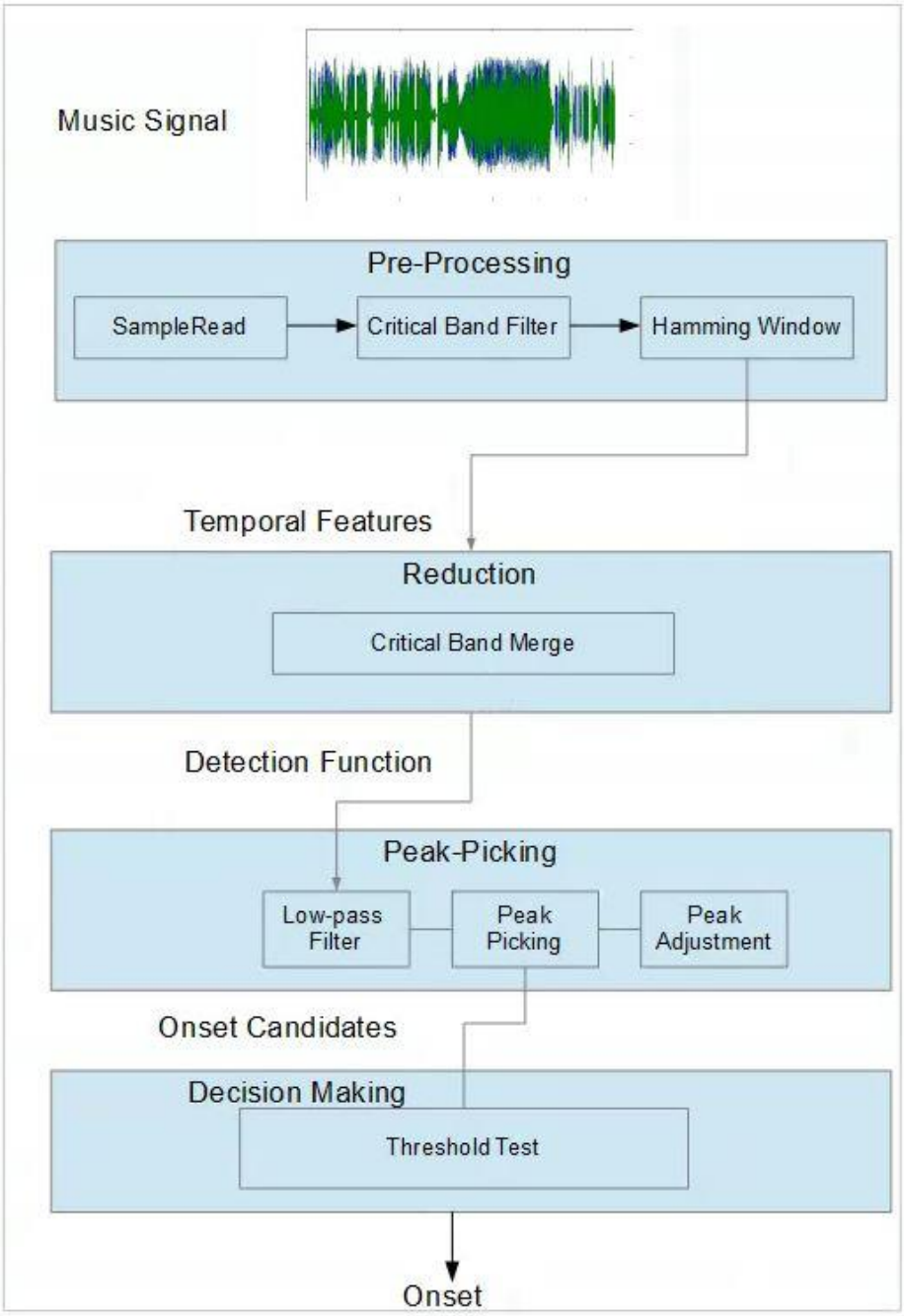

圖 2.2 時間特徵流程圖，取材自[1]

而在頻譜特偵偵測函數的部分，前處理部分都是將訊號以 512 個取樣點重疊一半的方式分為訊框，並乘上漢明窗後，將訊框以 FFT 轉換為頻域後，將得到的頻譜送入後續步驟處理。在頻譜特偵的降低維度階段中使用了三種方法：高頻內容、頻譜差值、上升頻帶數。

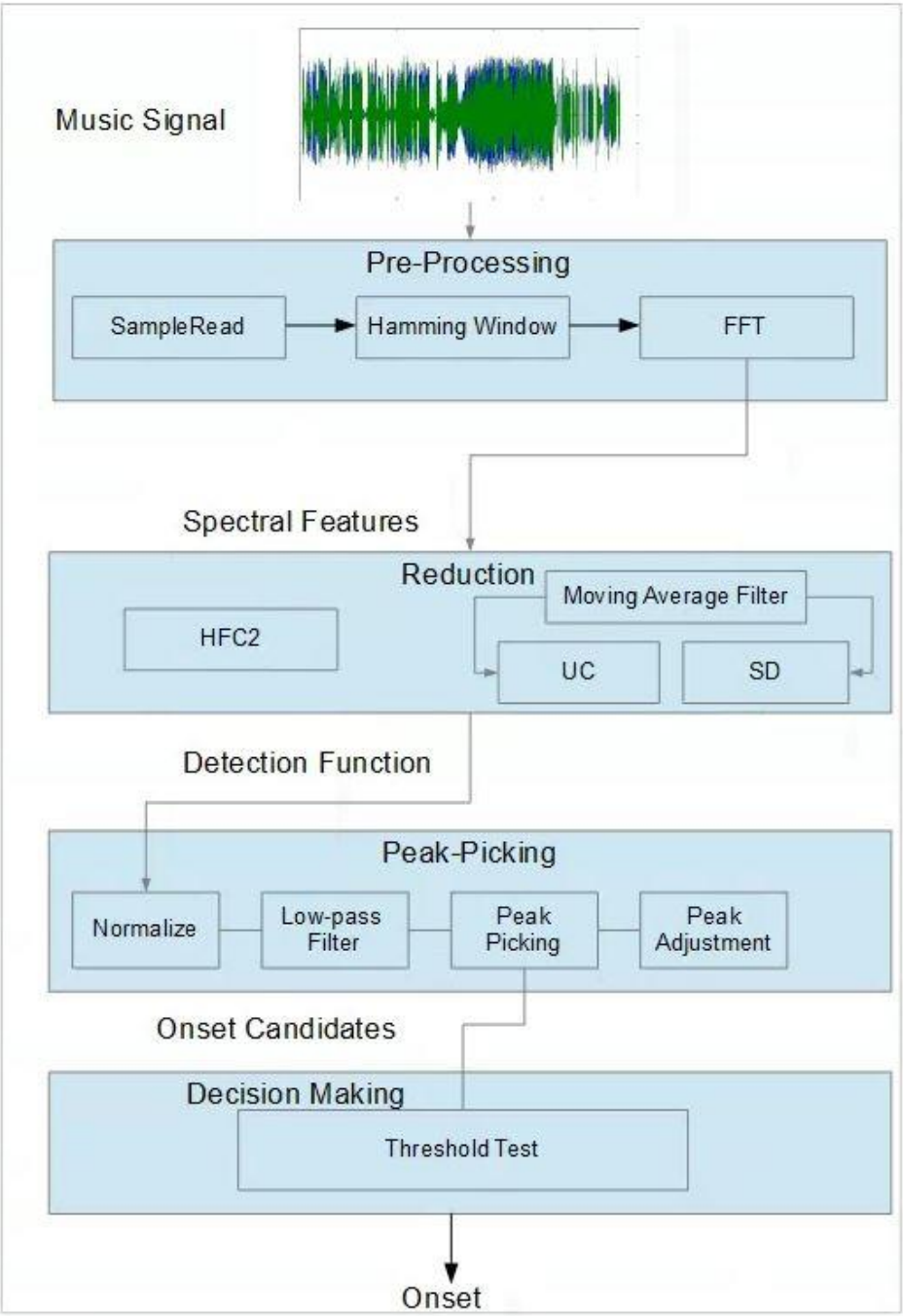

圖 2.3 頻率特徵流程圖，取材自[1]

第一種方法是高頻內容(High Frequency Content, HFC)[6]，利用起音點與高頻的能量變化有高度的相關性，也就是當音訊的高頻能量值變化越大，越有可能有起音點存在。因此引入了頻帶權重的概念，將音訊信號依據頻段給予加權，再將加權後頻帶能量加總，當總能量越大時，代表此訊框為起音點的可能性越高。計算如公式(2.2)。

$$HFC2(n) = \sum_{k=2}^{\frac{N}{2}} k^2 |X_k(n)|$$

(2.2)

其中$X_k(n)$為第 n 個訊框中第 k 個頻譜線(spectral line)的能量值，而k^2是所使用的權重，當 k 越大即越高頻時權重也越大。而用來做峰值擷取的特徵值是將兩相鄰訊框的 *HFC2* 值相減，其差值越高越有可能為起音點訊框，如公式(2.3)。

$$D_{HFC2}(n) = HFC2(n) - HFC2(n-1)$$

(2.3)

第二種方法是頻譜差值(Spectral Difference, SD)，是利用目前訊框與前一訊框在同樣第 k 個頻譜線能量差為基礎，當能量值改變越大時，目前訊框越有可能是起音點[6]，計算如公式(2.4)。

$$D_{SD}(n) = \sum_{k=1}^{N} H(|X_k(n)| - |X_k(n-1)|)^2$$

(2.4)

其中 H 如公式(2.5)。

$$H(x) = (x + |x|)/2$$

(2.5)

表示若 x 為正值則回傳其值，否則回傳 0。由公式可知當$D_{SD}(n)$越大時，代表該訊框能量上升越大，越有可能為起音點。

第三種方法是上升頻帶數(Up-Count, UC)[7]，類似於頻譜差值，但是僅計算能量上升的頻帶個數，而不加總上升的總能量，目的是避免特定頻帶上升能量的極大值主導整個訊框的能量變化值。計算如公式(2.6)。

$$D_{UC} = \sum_{k=1}^{N} G(|X_k(n)| - |X_k(n-1)|)$$

(2.6)

其中 *G(x)*表示當輸入的 x 值為正時回傳 1，否則回傳 0，是為了只計算能量上升頻代的個數。

2.2 librosa 與 madmom

因為在這次研究中，系統實作所使用的語言為 python，因此在挑選比較對象時，找了兩個在 python 上專門處理音訊信號的套件 librosa[2]與 madmom[3]。前者最早發表於 2015 年的 SciPy Conference，命名來自開發者當時所就讀的哥倫比亞大學語音和音頻識別和組織實驗室 LabROSA。而後者是由在音樂資訊檢索(music information retrieval,MIR)領域有諸多貢獻的奧地利林茲大學計算機認知系的研究者所開發。

2.2.1 librosa

librosa 中所提供的起音點偵測方式，用到了將音訊經短時距傅立葉變換(Short-time Fourier transform, STFT)轉換的時頻譜(spectrogram)，以及梅爾刻度(Mel scale)。

以 STFT 轉換的時頻譜，與使用 FFT 轉換的頻譜不同，除了頻率資訊外還保留了時間資訊，在分析上更能觀察出隨時間變化的狀態，一個由 librosa 繪製的時頻譜如圖 2.4。

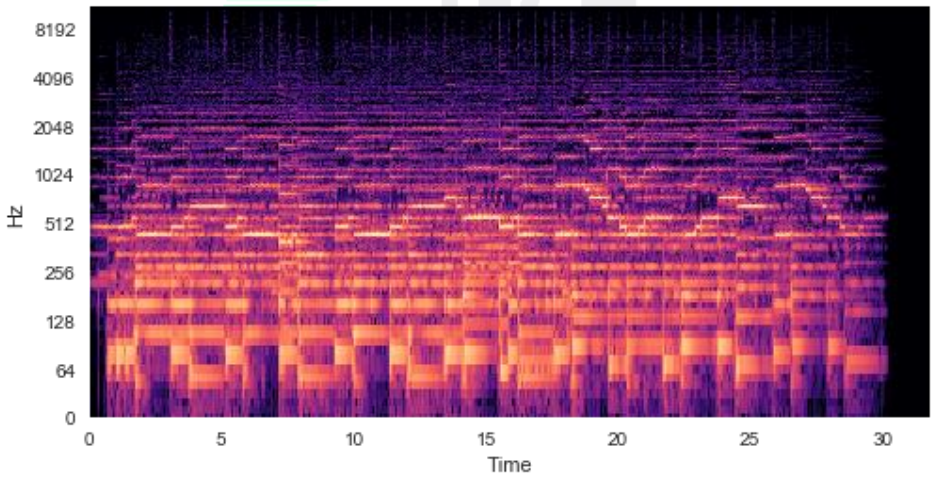

圖 2.4 以 librosa 繪製的時頻譜

而梅爾刻度，是一種以人耳對不同頻率聲音敏感度不同為基礎的一種非線性頻率刻度。人耳對於聲音的敏感度，在低頻變化的部分較敏感，對高頻變化較不敏感，因此經過梅爾刻度變換的頻率差更能貼近人耳對聲音的感受。一般頻率對梅爾頻率的曲線如圖 2.5。

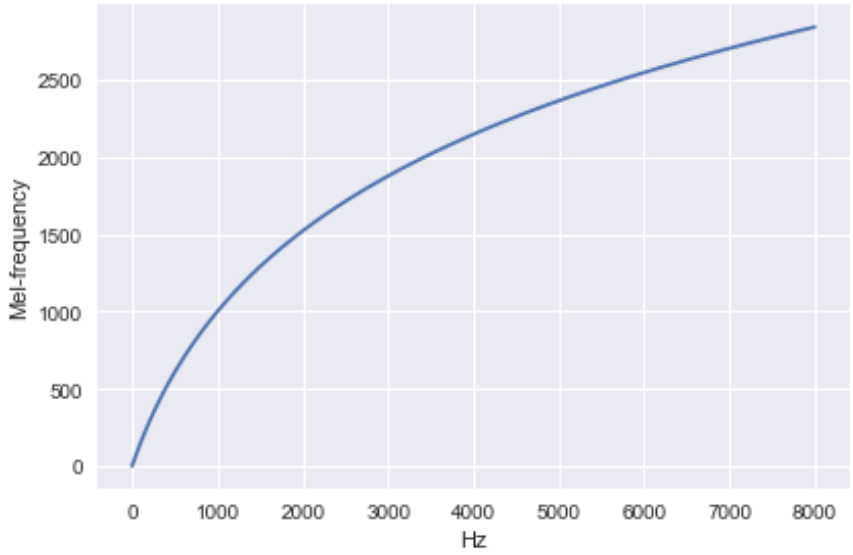

圖 2.5 一般頻率對梅爾頻率之曲線

librosa 的起音點偵測方式，是將音訊資料輸入，以 STFT 計算出時頻譜後，再經由梅爾濾波器轉換為梅爾刻度，接著取對數後計算每一訊框與前一訊框之能量差，去除負值後，對每個訊框上的不同頻帶之能量取平均值做為特徵值，最後對特徵值做峰值擷取即可得到可能的起音點位置。

2.2.2 madmom

madmom 中實作了多種起音點偵測方式，這次研究選擇了在其論文[3]中顯示結果最好的 CNNOnsetDetector。顧名思義，這種起音點偵測方式使用了卷積神經網絡(Convolutional Neural Network, CNN)，該方法發表在該套件作者中 Jan Schlüter 與 Sebastian Böck 所發表的論文[8]。

其概念是利用經過 STFT 變換的時頻譜與起音點所在位置的對應關係。因為起音點會出現在有明顯能量變化的地方，而在時頻譜上尋找能量變化明顯之處，就像在對圖形做邊緣偵測，因此嘗試訓練 CNN 來從時頻譜上找出可能是起音點的位置。

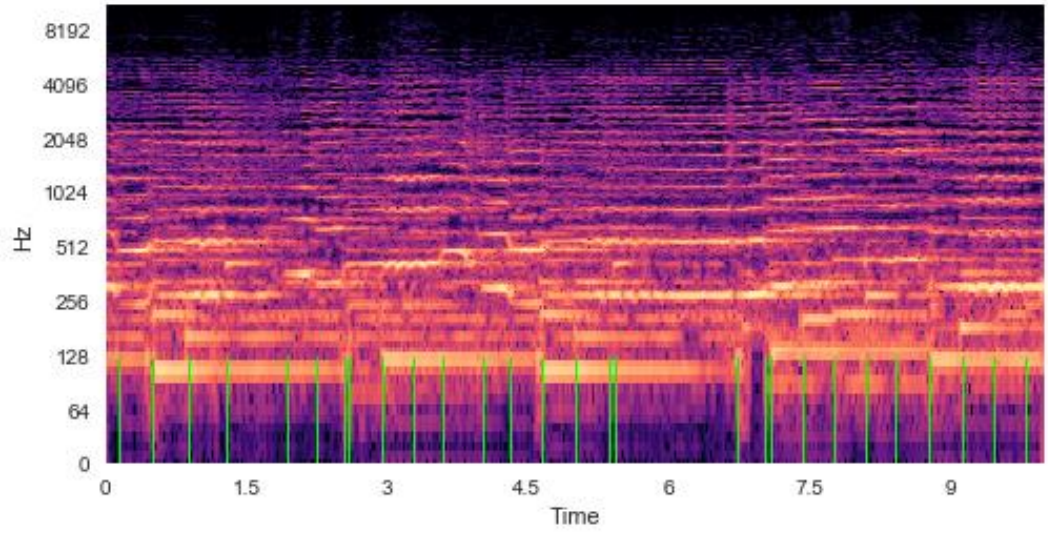

圖 2.6 一段樂曲的時頻譜與以綠線標記的起音點

其步驟是先將輸入樂曲經過訊框化、STFT 與梅爾刻度計算出梅爾時頻譜後，以訊框為單位輸入 CNN，使用的 CNN 架構是三個 input，經過卷積層(convolution layer)和 max-pooling 計算出 20 個 feature maps 後，以 full connected 的網路做分類，將每個訊框分類出是起音點(onsets)與非起音點(non-onsets)，輸出結果為該訊框是起音點的機率。最後對是起音點的機率的數列做峰值擷取得到起音點位置。

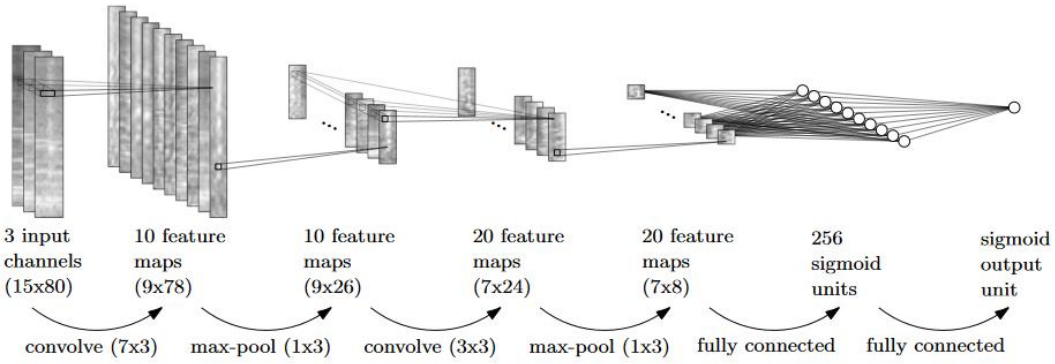

圖 2.7 madmom 的 CNN 網路架構，取材自[8]

2.3 實驗

2.3.1 起音點偵測正確率

為了進行起音點偵測正確率比較的實驗，使用了從奧地利林茲大學計算機認知系的 github 上所提供，起音點偵測研究用的 dataset，裡面包含了多組歌曲與紀錄其起音點位置的檔案，從中隨機選了 10 首歌曲，分別使用前述的 6 種方法計算起音點。

而在評估計算出來的起音點正確性方面，使用的測量方式是 F-measure，計算方式如公式(2.7)。

$$F = 2 \times \frac{P * R}{P + R} \tag{2.7}$$

其中 F 表示 F-measure，P 表示 Precision，R 表示 Recall。而 P 與 R 的計算如公式(2.8)與公式(2.9)。

$$P = \frac{O_{CD}}{O_{CD} + O_{FP}} \tag{2.8}$$

$$R = \frac{O_{CD}}{O_{CD} + O_{FN}} \tag{2.9}$$

比較 dataset 所提供的起音點位置，若有偵測到起音點於誤差時間(50ms)之內則標記為 correct detection (CD)，沒有偵測到則標記為 false negative (FN)，而在沒有起音點的位置偵測到起音點則標記為 false positive (FP)。O_{CD}、O_{FN}與O_{FP}分別表示對一首歌曲的起音點偵測中的 CD、FN 與 FP 的次數。

計算出來的 F-measure、Precision 與 Recall 都會介於 0 到 1 之間，數值越大代表正確性越高。其中 Precision 表示了在應該有起音點的位置，實際上有偵測到起音點的比例。Recall 表示了所有偵測出來的起音點中，實際上真的是起音點的比例。

測試結果分別紀錄在表 2.1、2.2、2.3、2.4、2.5、2.6 與 2.7。

表 2.1 時間特徵方式

	bandpass		
	Precision	Recall	F-Measure
歌曲 1	0.481481	0.464286	0.47272727
歌曲 2	0.5	0.375	0.42857143
歌曲 3	0.206897	0.285714	0.24
歌曲 4	0.642857	0.375	0.47368421
歌曲 5	0.972973	0.666667	0.79120879
歌曲 6	0.903226	0.430769	0.58333333
歌曲 7	0.75	0.418605	0.53731343
歌曲 8	0.541667	0.209677	0.30232558
歌曲 9	0.5	0.464286	0.48148148
歌曲 10	1	0.722222	0.83870968
Average	0.64991	0.441223	0.51493552

表 2.2 高頻內容

	HFC		
	Precision	Recall	F-Measure
歌曲 1	0.209677	0.928571	0.34210526
歌曲 2	0.362637	0.825	0.50381679
歌曲 3	0.4	0.47619	0.43478261
歌曲 4	0.657895	0.520833	0.58139535
歌曲 5	0.607595	0.888889	0.72180451
歌曲 6	0.716981	0.584615	0.6440678
歌曲 7	0.568966	0.767442	0.65346535
歌曲 8	0.553571	0.5	0.52542373
歌曲 9	0.318182	0.75	0.44680851
歌曲 10	0.538462	0.583333	0.56
Average	0.493397	0.682487	0.54136699

表 2.3 頻譜差值

	SD		
	Precision	Recall	F-Measure
歌曲 1	0.307692	0.428571	0.35820896
歌曲 2	0.705882	0.3	0.42105263
歌曲 3	0.311111	0.666667	0.42424242
歌曲 4	0.894737	0.354167	0.50746269
歌曲 5	1	0.185185	0.3125
歌曲 6	0.527778	0.292308	0.37623762
歌曲 7	0.842105	0.372093	0.51612903
歌曲 8	0.842105	0.258065	0.39506173
歌曲 9	0.533333	0.285714	0.37209302
歌曲 10	1	0.138889	0.24390244
Average	0.696474	0.328166	0.39268905

表 2.4 上升頻帶數

	UC		
	Precision	Recall	F-Measure
歌曲 1	0.247706	0.964286	0.39416058
歌曲 2	0.323232	0.8	0.46043165
歌曲 3	0.171171	0.904762	0.28787879
歌曲 4	0.38	0.791667	0.51351351
歌曲 5	0.8125	0.722222	0.76470588
歌曲 6	0.926829	0.584615	0.71698113
歌曲 7	0.666667	0.651163	0.65882353
歌曲 8	0.661765	0.725806	0.69230769
歌曲 9	0.2875	0.821429	0.42592593
歌曲 10	0.609756	0.694444	0.64935065
Average	0.508713	0.766039	0.55640794

表 2.5 librosa

	librosa		
	Precision	Recall	F-Measure
歌曲 1	0.324675	0.892857	0.47619048
歌曲 2	0.775	0.775	0.775
歌曲 3	0.269231	1	0.42424242
歌曲 4	0.6	0.75	0.66666667
歌曲 5	1	0.759259	0.86315789
歌曲 6	1	0.6	0.75
歌曲 7	0.911765	0.72093	0.80519481
歌曲 8	0.886364	0.629032	0.73584906
歌曲 9	0.571429	0.857143	0.68571429
歌曲 10	1	0.805556	0.89230769
Average	0.733846	0.778978	0.70743233

表 2.6 madmom

	madmom		
	Precision	Recall	F-Measure
歌曲 1	0.875	0.75	0.80769231
歌曲 2	0.96	0.6	0.73846154
歌曲 3	1	0.714286	0.83333333
歌曲 4	1	0.541667	0.7027027
歌曲 5	1	0.814815	0.89795918
歌曲 6	1	0.938462	0.96825397
歌曲 7	1	0.72093	0.83783784
歌曲 8	1	0.693548	0.81904762
歌曲 9	0.954545	0.75	0.84
歌曲 10	1	0.833333	0.90909091
Average	0.978955	0.735704	0.83543794

表 2.7 以上 6 種方法的平均值

Average			
	Precision	Recall	F-Measure
bandpass	0.64991	0.441223	0.51493552
HFC	0.493397	0.682487	0.54136699
SD	0.696474	0.328166	0.39268905
UC	0.508713	0.766039	0.55640794
librosa	0.733846	0.778978	0.70743233
madmom	0.978955	0.735704	0.83543794

從測試結果可以觀察到前研究所使用的各種方法有一定程度的正確率，librosa 的正確率比前研究的方法都還要更好，而 madmom 的正確率則又比 librosa 還要來得更好一些。

2.3.2 起音點正確率對歌曲相似度比較之影響

在前一節中，我們比對了前研究與兩種套件所提供之起音點偵測方法的正確率，而其中雖然 madmom 的起音點偵測方法有最高的正確率，但是在我們的實驗過程中，發現部分歌曲在以 madmom 進行起音點偵測時會出現程式錯誤，因此在這一階段中，我們是選擇 librosa 與前研究的起音點偵測方法，來實驗起音點偵測的正確率對歌曲相似度比較的影響。

這邊所使用的歌曲相似度比較是前研究所使用的方法，首先在之前的步驟中得到起音點之後，以 RLCS 兩兩計算，比對歌曲的起音點數列計算出用來評價相似度的分數。在比對歌曲相似度方面，我們準備了一個包含 1966 首各種不同風格類型歌曲的資料庫，並隨機選了 5 首做為比較基準的歌曲。

在 liborosa 部分，以 librosa 中的起音點偵測計算所有歌曲的起音點，以 RLCS 算出 5 首基準歌曲與其他所有歌曲的相似度分數。

而在前研究的方法中，首先以前面所述的 4 種方法分別計算所有歌曲的起音點，接著分別計算出 RLCS 分數，並且為了提升準確性，將 4 種方法計算出來的 RLCS 分數，個別乘上權重後加總得到單一分數，所使用的權重分別是時間特徵方式 0.4、高頻內容 0.2、頻譜差值 0.2、上升頻帶數 0.2。

在所有分數計算完畢後，分別取與各首基準歌曲相似度分數前 10 高的歌曲，來觀察起音點偵測準確性較高的 librosa，與起音點偵測準確性較為普通的前研究方法，所找出的最相似歌曲是否會有很大的區別。計算出的測試結果如表 2.7。

從結果中可以看出，其中三首基準歌曲所找出的相似歌曲，兩方法找出的結果有較多的重疊，另外兩首則重疊性較低。而重疊性較低的兩首基準歌曲，可以發現到 librosa 所計算出的最高相似分數明顯低於另外三首，我們認為是因為這兩首歌曲與資料庫中的其他歌曲相似度都不高。因此儘管前研究的方法起音點偵測準確度較差，但是在做相似度比對時，與起音點偵測準確度較高的方法比對，還是能發揮不差的效果。原因可能是因為在較低準確性的方法中，發生 false negative 與 false positive 的位置也存在著相似性。但是另一方面，起音點偵測準確度較高的方法所計算出的分數，在歌曲相似度都偏低時，能提供較明顯的鑑別度。

表 2.8(a) 以 RLCS 計算相似度分數的結果，沒有分數的表示是比較基準歌曲，上了相同底色的部分表示在兩種方法中都有找到相同歌曲

librosa		MUX	
score	name	score	name
	Allrise01.wav		Allrise01.wav
0.579103046	合奏專修1-12.wav	0.354758516	英文童謠6-21.wav
0.55913859	兒童班3Let's Have Fun Together-6.wav	0.345960119	合奏專修1-12.wav
0.556879057	WhereIsTheLove01.wav	0.342579657	OutsideCastle07.wav
0.55177703	OutsideCastle07.wav	0.33915309	original soundtrack1CD13.wav
0.55177703	英文童謠6-21.wav	0.338223702	校園民歌精選3-2.wav
0.550284238	校園民歌精選4-6.wav	0.335639527	FROSTY THE SNOWMAN-10.wav
0.550249545	great classical marches-7.wav	0.330239377	WhereIsTheLove01.wav
0.55	先修班4-22.wav	0.327637063	老式情歌-9.wav
0.547189436	校園民歌精選3-2.wav	0.324521622	校園民歌精選4-6.wav
0.543888889	Concsrto for Clarinet and Orchestra05.wav	0.324434349	vivaldi the four seasons-7.wav
	BEST en route01.wav		BEST en route01.wav
0.376792115	百合二重唱_鄉土追想1-8.wav	0.298021374	侍09.wav
0.37165404	masters of classical music vol.6 peter tchaikovsky-4.wav	0.286412589	together-8.wav
0.358225176	original soundtrack1CD06.wav	0.286338511	彩-5.wav
0.357878352	barry tuckwell english chamber orchestra-10.wav	0.284632891	兒童班2Delicious Dream-11.wav
0.350505191	PRAHA DANCES-6.wav	0.284053452	金片子二07.wav
0.350505191	愛唱歌精選02.wav	0.281714071	Viennese Violin-2.wav
0.34785638	THE CONCERT IN CENTRAL PARK-3.wav	0.279635091	DANCE05.wav
0.345766129	mana mouskouri-13.wav	0.276738364	The Firebird13.wav
0.345766129	ANNE MURRAY ALL-TME Greatest Hits Volume 3-3.wav	0.275480545	校園民歌精選2-3.wav
0.343318966	金片子二07.wav	0.270456207	masters of classical music vol.6 peter tchaikovsky-9.wav
	Celtic Moon01.wav		Celtic Moon01.wav
0.54788961	兒童班1let's go to the Zoo-2.wav	0.342332425	兒童班1let's go to the Zoo-2.wav
0.433837535	校園民歌精選2-4.wav	0.287820423	成名金曲精選09.wav
0.431941569	愛唱歌精選08.wav	0.282161382	成名金曲精選14.wav
0.431746032	Allrise11.wav	0.280586413	mana mouskouri-1.wav
0.412885154	校園民歌精選3-7.wav	0.272899585	Queens of the Stone Age02.wav
0.41025641	先修班2-5.wav	0.270450643	台灣歌聲10.wav
0.407498184	成名金曲精選09.wav	0.269590168	成名金曲精選05.wav
0.40707483	susanna sharpe and samba police-4.wav	0.269174559	海波浪06.wav
0.402148787	合奏專修2-1.wav	0.265853579	校園民歌精選2-4.wav
0.39449113	original soundtrack2CD10.wav	0.260510085	天誅06.wav
	original soundtrack1CD01.wav		original soundtrack1CD01.wav
0.37214886	侍20.wav	0.346648885	together-8.wav
0.371822803	PRAHA DANCES-27.wav	0.343737074	The Firebird13.wav
0.370588235	Madonna11.wav	0.335699672	masters of classical music vol.6 peter tchaikovsky-9.wav
0.360294118	在台紀念CD01.wav	0.33380125	enya watermark-10.wav
0.347432306	天誅13.wav	0.324394927	cd書15.wav
0.347432306	enya watermark-12.wav	0.320412564	校園民歌精選2-3.wav
0.343045535	SCOTTISH CHAMBER ORCHESTRA04.wav	0.319363273	together-6.wav
0.3412274	new york philharmonic zubin mehta-2.wav	0.318605615	咱的歌-7.wav
0.3412274	REFLECTION OF my life 4-12.wav	0.315551581	ANNE MURRAY ALL-TME Greatest Hits Volume 2-3.wav
0.341176471	Celtic Moon14.wav	0.310502488	兒童班2Delicious Dream-11.wav
	成名金曲精選01.wav		成名金曲精選01.wav
0.640540995	Allrise11.wav	0.385614312	整個八月09.wav
0.613351254	整個八月09.wav	0.379075389	original soundtrack2CD10.wav
0.596886201	海波浪05.wav	0.370553541	海波浪05.wav
0.580645161	antonio vivaldi10.wav	0.35231609	original soundtrack1CD03.wav
0.55239899	兒童班1let's go to the Zoo-2.wav	0.345311204	英文童謠3-20.wav
0.533266129	Encores06.wav	0.342387463	antonio vivaldi10.wav
0.533266129	original soundtrack2CD10.wav	0.340630352	Allrise11.wav
0.532203758	校園民歌精選3-7.wav	0.32845049	NANA MOUSKOURI-5.wav
0.524933846	英文童謠6-12.wav	0.326482492	兒童班3Let's Have Fun Together-8.wav
0.517921147	英文童謠6-7.wav	0.31944	BEST en route12.wav

表 2.8(b) (a)中的相同歌曲數

	librosa 與前研究方法相同歌曲數
基準歌曲 1	6
基準歌曲 2	1
基準歌曲 3	3
基準歌曲 4	0
基準歌曲 5	5

第三章 改進 RLCS 計算效率

在前一章中，我們提到了用來計算兩個起音點數列相似度的 RLCS 演算法，在這一章中將簡單介紹 RLCS 演算法的計算流程，並說明為了改進 RLCS 計算效率，使用了那些方法。

3.1 RLCS

以下簡單介紹 RLCS 演算法[10]，最長共同子序列(Longest Common Subsequence, LCS)是用來找出兩個序列之間最長的共同部分的演算法，假設有兩序列A = $\{a_1, \dots, a_i\}$、B = $\{b_1, \dots, b_j\}$，其計算方式如公式(3.1)。

$$LCS[i,j] = \begin{cases} 0 & ,i = 0 \text{ or } j = 0 \\ LCS[i-1, j-1] + 1 & ,i \times j > 0 \text{ and } a_i = b_j \\ \max(LCS[i, j-1], LCS[i-1, j]) & ,i \times j > 0 \text{ and } a_i \neq b_j \end{cases} \tag{3.1}$$

假設分別有序列 A={13, 1, 20, 8, 5, 13, 1, 20, 9, 3}、A'={13, 1, 1, 19, 20, 18, 9, 3, 8, 20}與 B={13, 1, 20, 8}，我們可以計算其 LCS 長度皆為 4，但是其共同部分的跨度不同，如圖 3.1、圖 3.2。其中序列 A 和 B 的 LCS 跨度 4 較小，相似度比跨度 9 的序列 A'和 B 更高，但是在 LCS 中無法表現出來。

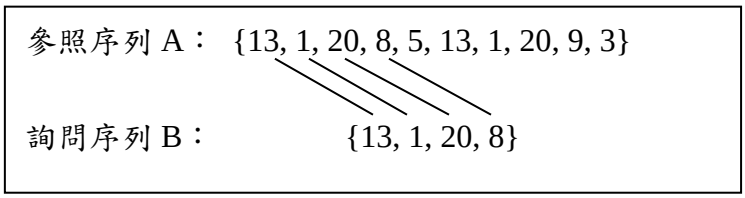

圖 3.1 LCS 跨度較小

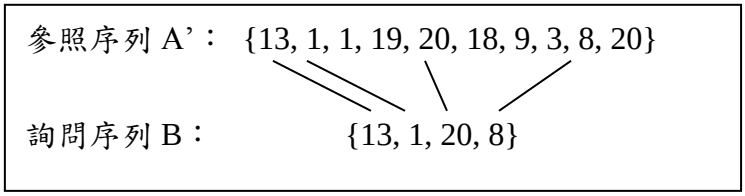

圖 3.2 LCS 跨度較大

為了解決這個問題，RLCS 中加入考慮跨度的參數，width across reference(WAR)和 width across query(WAQ)，分別代表共同序列在參照序列和詢問序列中的跨度，其計算方式如公式(3.2)、(3.3)。

$$W_R[i,j]=\begin{cases}0 & ,i\times j=0\\ W_R[i-1,j-1] & ,i\times j>0,a_i=b_j\\ W_R[i-1,j]+1 & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]\geq LCS[i,j-1],W_R[i-1,j]>0\\ 0 & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]\geq LCS[i,j-1],W_R[i-1,j]=0\\ W_R[i,j-1] & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]<LCS[i,j-1]\end{cases} \tag{3.2}$$

$$W_Q[i,j]=\begin{cases}0 & ,i\times j=0\\ W_Q[i-1,j-1]+1 & ,i\times j>0,a_i=b_j\\ W_Q[i-1,j] & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]\geq LCS[i,j-1]\\ W_Q[i,j-1]+1 & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]<LCS[i,j-1],W_Q[i,j-1]>0\\ 0 & ,i\times j>0,a_i\neq b_j,LCS[i-1,j]<LCS[i,j-1],W_Q[i,j-1]=0\end{cases} \tag{3.3}$$

接著計算出分數的陣列如公式(3.4)。

$$score[i,j]=\begin{cases}\left(\alpha\dfrac{LCS[i,j]}{w^R[i,j]}+(1-\alpha)\dfrac{LCS[i,j]}{w^Q[i,j]}\right)\times\dfrac{LCS[i,j]}{N} & ,if\ LCS[i,j]\geq\lambda N\\ 0 & ,else\end{cases} \tag{3.4}$$

最後取陣列中的最大值即為兩序列間的 RLCS 分數。假設兩序列長分別為 M、N，從計算公式可以看出 RLCS 在運算過程需要跑$M\times N$次的迴圈，當起音點數量以及需要比對的歌曲數量龐大時，在計算時將會耗費非常龐大的時間。因此，希望能用一些方法來改善計算的效率。

3.2 Cython

因為這次研究的實作是以 Python 來寫成，在考慮加速運算的方法時首先想到了 Cython[11]。Cython 是為了能夠輕易地製作 Python 的 C 語言擴張模組所開發的語言，幾乎完全相容原有的 Python，能夠將 Python 原始碼轉換為 C 的程式碼，並編譯成 Python 的擴張模組。

由於 Python 本身是直譯式語言，在執行效率相比編譯式語言差上不少。因此很多 Python 的模組或套件為了能有良好的運算效率，是以 C 語言寫成，並編譯為 Python 可用的模組。而 Cython 能讓使用者幾乎不需要更改原始的

Python 程式碼，就能轉換並編譯出 Python 的 C 語言擴張模組，提供簡單提升計算效率的方式。

除了直接原封不動的進行轉換之外，Cython 也有更多的深入的方式能進一步提升計算效率。這裡實作使用了一個最常用的方法，就是指定 Python 中變數的變數型態。在 Python 中，變數是不需要指定型態的，變數的型態會在執行時才決定，這種作法能提升開發效率，但是也會讓計算效率下降不少。而在 Cython 中，可以預先指定變數型態，讓程式執行時不需要再進行判斷，能夠進一步的提升計算效率。

舉例來說，下面是 RLCS 中宣告一個M × N大小的陣列的程式碼：

```
M = len(database)
N = len(query)
C = np.zeros((M+1, N+1))
```

在以 Cython 指定變數型態後的程式碼如下：

```
cdef int M = len(database)
cdef int N = len(query)
cdef np.ndarray[FTYPE_t, ndim=2] C = np.zeros((M+1, N+1), dtype=FTYPE)
```

M 和 N 是整數型態的 int，而 C 是一個 Numpy 的二維陣列，陣列元素是 Numpy 中的浮點數(以 ctypedef np.float_t FTYPE_t 定義)。

在本次實作中，將用來計算 RLCS 的函數抽出，以 Cython 改寫並編譯成 pyd 檔(以 Python 產生的 Windows 動態連結庫)，再從原程式中匯入(import)並呼叫該函數。

3.3 平行處理

在每次有要詢問的歌曲輸入系統中，要比對歌曲的相似度就必須與系統中所有的歌曲比較。而在實際運用上，資料庫中包含的歌曲數可能有上萬首，需要進行非常多次的 RLCS 運算。而與不同首歌曲比較時，因為彼此之間的運算沒有任何相依性，所以能夠簡單的以平行運算的方式，把與不同歌曲的比較，以不同的程序(process)來執行。

在這裡所使用的是 Python 內建的模組 multiprocessing 中的 pool，宣告 pool 並指定要使用的子程序數後，以 pool 中的 starmap 函數把所有要進行比對的歌曲組合陣列配對上 RLCS 函數，就可以把每一次函數執行配對到產生的子程序上。

實驗所使用的環境是四核心八執行緒的 CPU，為了避免系統滿載，在實作上 pool 所使用的是 6 個子程序。

3.4 效率測試

為了測試上述的方法實際上加快了多少執行速度，準備了 4 個版本的程式來做比較。分別是原始沒有使用任何方法加速的版本、只使用 Cython 的版本、只使用平行處理的版本、同時使用 Cython 與平行處理的版本。

而測試所使用的資料是一個有 1966 首歌曲起音點的資料庫，每首歌曲的長度為 15 秒。測試過程是將第一首歌曲與其他的 1965 首歌曲的起音點做比對，也就是說總共需要進行 1965 次的 RLCS 計算來得到所有分數。

表 3.1 執行時間比較

版本	執行時間(秒)
原始版本	41.96
Cython	13.06
平行處理	11.28
Cython+平行處理	4.23

表 3.2 以原始版本為基準的加速比例

版本	加速比例
原始版本	1.00
Cython	3.21
平行處理	3.72
Cython+平行處理	9.92

實際執行的測試結果如表 3.1、3.2 所示。可以看到不論是單獨使用 Cython 或是平行處理的方式，加速的比例都有達到 3 倍以上，同時使用兩種方法更是加速將近 10 倍。其中平行處理的部分，雖然是以 6 個子程序執行，但是子程序每次計算必須與主程序溝通，從主程序取得需要計算的資料，計算完畢後將結果回傳給主程序，因此實際上的加速沒辦法到 6 倍那麼多。

以一個歌曲推薦系統來說，當使用者輸入歌曲後，如果需要太長時間的計算會容易讓人等得不耐煩，在經過本章所使用的方法加速後，能夠比較符合一個推薦系統可實用化的執行效率。而在實驗研究方面，也大幅減少了在反覆測試時所需要等待的時間，能夠讓實驗過程變得更有效率。本章所進行的改良與測試，有達到了預期的成果。

第四章 歌曲推薦系統

本章中將對前幾章中所述的前研究方法整合和成的歌曲推薦系統，再次進行簡單的統整性描述，並進行與其他歌曲推薦系統的比較。做為比較對象的系統，是歌曲相似度比較函式庫的 msuly，在這章中會簡述該函式庫所使用的方法。而比較的方式是以實際人耳所感受到的相似與否來進行的聆聽實驗。

4.1 前研究

如前幾章所述，前研究所使用的方法是以起音點偵測為基礎，以一種時間特徵的方式，與三種頻譜特徵的方式分別計算出起音點。並以 RLCS 演算法分別計算出詢問歌曲與參照歌曲間的四種相似度分數，最後將四個分數以時間特徵方法 0.4，三種頻譜特徵方法各 0.2 的權重進行加總，得到兩首歌曲間的相似度。

在將詢問歌曲與所有參照歌曲比對計算完畢後，將得到的所有相似度分數與對應的歌曲以分數由高至低進行排序，並取相似度分數前幾高的歌曲來做為推薦歌曲。系統架構如圖 4.1。

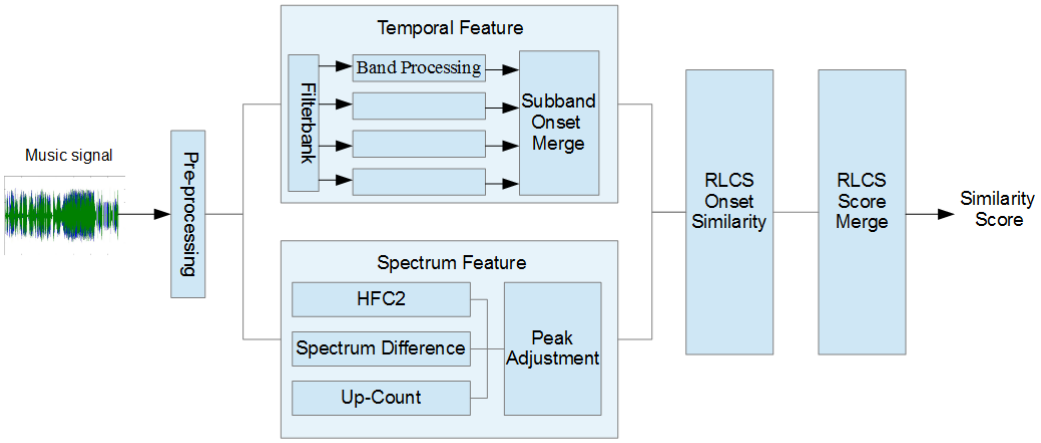

圖 4.1 前研究方法之系統架構

4.2 musly

musly 是一個以 C/C++寫成的歌曲相似度比較函式庫[4]。該函式庫實作了兩種歌曲相似度演算法，而第三種商業用的方法需要向 OFAI 申請授權來取得額外的外掛程式，因此沒有使用在本研究中。以下將簡單介紹兩種演算法。

4.2.1 Mandel-Ellis

這個演算法是由 M. Mandel 和 D. Ellis 於 2005 年在 ISMIR 所發表的論文[12]中所提出的。首先以梅爾倒頻譜係數(Mel-frequency cepstral coefficients (MFCC)來抽取特爭值，這邊所使用的是 20 個係數的 MFCC，所抽取出來的特徵被描述為是音色的特徵。並將每首歌曲的特徵值建立為單一高斯模型(single Gaussian model)，之後以 KL 散度(Kullback–Leibler divergence，又稱為相對熵 relative entropy)對建立出來的模型兩兩進行比較，來得到不同模型間的距離來衡量歌曲間的相似度。假設有兩機率分布 $p(x)$、$q(x)$，KL 散度的定義如公式(4.1)。

$$KL(p||q) = \int p(x) \ln \frac{p(x)}{q(x)} dx \tag{4.1}$$

4.2.2 Timbre

第二個演算法是 musly 以上述的 Mandel-Ellis 演算法為基礎，並加以改良來得到更好的結果，命名為 Timbre，即音色之意。其中 MFCC 所使用的係數是 25 個，並使用以 KL 散度為基礎改良的 JS 散度(Jensen–Shannon divergence)。同樣假設有兩機率分布 p、q，JS 散度定義如公式(4.2)。

$$JS(p||q) = \frac{1}{2} KL(p||m) + \frac{1}{2} KL(q||m) \tag{4.2}$$

其中 m 如公式(4.3)：

$$m = \frac{1}{2}(p + q) \tag{4.3}$$

此外，對相似度以 Mutual Proximity 做一般化。Mutual Proximity 是由 musly 開發者 Dominik Schnitzer 與另外三位作者在 2011 年的 ISMIR 發表的論文[13]中所提出的。Mutual Proximity 的做法如下，首先將前步驟中所計算出的所有距離計算其平均值與標準差，以建立出機率分布 X，以此步驟將兩點 x 與 y 的距離$d_{x,y}$轉換為 y 是最接近 x 的鄰居的機率，其機率表現為在分布 X 中的$P(X > d_{x,y})$。之後將 y 是最接近 x 的鄰居的機率與 x 是最接近 y 的鄰居的機率相乘得到兩點的 Mutual Proximity，如公式(4.4)。

$$MP(d_{x,y}) = P(X > d_{x,y}) \times P(X > d_{y,x}) \tag{4.4}$$

由公式可知，Mutual Proximity 所計算出來的結果是具有對稱性的，即 $MP(d_{x,y}) = MP(d_{y,x})$，並且數值被一般化為介於 0 到 1 之間，可以將原本散度計算出的非對稱的距離變為對稱的近似機率。Timbre 為 musly 預設使用的演算法，因此本實驗也選擇使用該演算法。

4.3聆聽實驗

為了比較上述兩套系統，誰比較接近人耳所感到是否相似，我們準備了聆聽實驗。實驗所使用的資料，是前面也有使用到的，包含有 1966 首各種不同曲風歌曲的資料庫，從中隨機選出 5 首歌曲作為比較基準，分別輸入兩套系統，得到這 5 首歌曲與其他歌曲的相似度，並分別挑出與 5 首歌曲相似度前 3 名的歌曲來做為比較對象。

我們總共找來了 15 名受測者，受測者會拿到兩個資料夾 musly 與 onset RLCS(前研究方法)，其下各有 5 個資料夾，分別放了 5 首基準歌曲與各自 3 首的比較對象歌曲，檔名皆重新命名為單純的編號。

讓受測者聆聽比較的過程分為兩個階段，第一階段先從各資料夾的 3 首比較對象中選出與基準歌曲最相似的歌曲。接著在第二階段，對於同一首基準歌曲，在經過第一階段會留下 musly 與 onset RLCS 各自最相似的歌曲，受測者要再從這兩首歌曲中選擇何者與基準歌曲更為相似，並回答認為該歌曲更為相似的原因。

相似的原因有幾個選項可以從中複選，分別是歌曲風格相似、歌曲旋律相似、歌曲節奏/速度相似、樂器配置相似、歌手音色相似或相同歌手、其他：請敘述。實驗結果如表(4.1)、(4.2)、(4.3)、(4.4)、(4.5)、(4.6)。

表 4.1 onset RLCS 選出的最相似歌曲前 3 名

onset RLCS	
score	name
	Allrise01.wav
0.354758516	(3)英文童謠 6-21.wav
0.345960119	(2)合奏專修 1-12.wav
0.342579657	(1)OutsideCastle07.wav
	BEST en route01.wav
0.298021374	(2)侍 09.wav
0.286412589	(1)together-8.wav
0.286338511	(3)彩-5.wav
	Celtic Moon01.wav
0.342332425	(3)兒童班 1let's go to the Zoo-2.wav
0.287820423	(1)成名金曲精選 09.wav
0.282161382	(2)成名金曲精選 14.wav
	original soundtrack1CD01.wav
0.346648885	(3)together-8.wav
0.343737074	(2)The Firebird13.wav
0.335699672	(1)masters of classical music vol.6 peter tchaikovsky-9.wav
	成名金曲精選 01.wav
0.385614312	(3)整個八月 09.wav
0.379075389	(1)original soundtrack2CD10.wav
0.370553541	(2)海波浪 05.wav

表 4.2 musly 選出的最相似歌曲前 3 名

musly	
Q/R	name
	Allrise01.wav
0.028036	(1)唱歌的人 08.wav
0.0284417	(3)天誅 04.wav
0.0292444	(2)生命之歌 01.wav
	BEST en route01.wav
0.0913413	(1)BEST en route13.wav
0.134997	(2)EXPERiENCE THE DiViNE BETTE MiDLER-10.wav
0.149597	(3)愛唱歌精選 03.wav
	Celtic Moon01.wav
0.0189399	(2)醉人的世界名曲 2-10.wav
0.0287417	(1)合奏專修 1-8.wav
0.0293173	(3)醉人的世界名曲 2-11.wav
	original soundtrack1CD01.wav
0.0362047	(3)校園民歌精選 4-4.wav
0.0804083	(2)愛唱歌精選 11.wav
0.0863408	(1)大地之愛-13.wav
	成名金曲精選 01.wav
0.0145523	(1)barry tuckwell english chamber orchestra-1.wav
0.0546112	(3)兒童班 1let's go to the Zoo-15.wav
0.0675374	(2)James Galway-1.wav

表 4.3 onset RLCS 最相似歌曲

	onset RLCS				
受測者	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
1	1	1	3	3	3
2	1	1	1	3	3
3	1	1	1	3	2
4	1	1	1	3	3
5	1	1	2	3	3
6	1	1	1	3	2
7	1	1	1	3	3
8	1	2	3	3	2
9	1	2	1	3	3
10	1	1	1	3	3
11	1	1	3	3	3
12	1	2	1	3	3
13	1	2	3	3	3
14	1	2	1	3	3
15	1	1	1	3	3

表 4.4 musly 最相似歌曲

	musly				
受測者	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
1	3	1	1	1	1
2	3	3	3	3	2
3	3	1	1	3	1
4	3	1	1	3	2
5	3	1	3	1	1
6	2	1	1	1	1
7	3	1	1	3	2
8	2	1	1	3	2
9	3	2	1	3	1
10	2	1	3	3	1
11	1	1	1	3	2
12	1	1	2	3	3
13	2	3	1	2	2
14	2	1	1	2	2
15	2	1	1	3	2

表 4.5(a) 何者較相似，其中 O 表示 onset RLCS 較相似

M 表示 musly 較相似，B 表示兩者都相似

N 表示兩者都不相似

	compare				
受測者	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
1	O	M	M	N	O
2	O	M	M	M	O
3	N	M	M	M	N
4	N	M	M	M	O
5	O	M	M	O	O
6	O	O	M	M	O
7	O	M	M	B	O
8	O	M	M	M	O
9	O	M	M	M	O
10	O	M	M	M	O
11	O	M	M	M	O
12	M	O	O	O	M
13	O	M	M	M	O
14	O	M	M	M	O
15	M	M	M	M	O

表 4.5(b) (a)之各選項數量整理

	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
onset RLCS 較相似	11	2	1	2	13
musly 較相似	2	13	14	11	1
兩者都相似	0	0	0	1	0
兩者都不相似	2	0	0	1	1

表 4.6(a) 受測者所填寫的相似原因

	reason				
受測者	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
1	歌曲節奏相似	歌手音色相似	樂器配置相似		歌曲節奏相似
2	歌曲節奏相似	樂器配置相似	樂器配置相似	歌曲旋律相似	歌曲節奏相似
3		歌手音色相似	樂器配置相似	歌曲旋律相似	
4		歌手音色相似	樂器配置相似	歌曲旋律相似	歌曲節奏相似
5	歌曲節奏相似	歌曲風格相似、歌曲旋律相似	歌曲風格相似、歌曲旋律相似	歌曲風格相似、歌曲旋律相似	歌曲節奏相似
6	歌曲風格、旋律	旋律	樂器配置	節奏	節奏
7	風格相似/速度相似	音色相似	樂器相似	旋律相似/風格相似	旋律相似
8	風格較相近	風格較相近	開頭較相近	開頭較相近	風格較相近
9	節奏比較像	旋律比較接近	選擇的這個沒有人聲	選擇的這個有人聲	節奏比較輕快
10	重低音的感覺很像	歌手聲音很像	節奏和樂器很像	背景音色相似	節奏和樂器很像
11	歌手聲音較像	歌手聲音較像	樂器較像	音調較像	節奏較像
12	風格較相近	風格較相近	樂器較相近	風格較相近	風格較相近
13	歌曲節奏/速度相似、歌曲風格相似	風格較相近	樂器較相近	風格較相近	歌曲節奏/速度、歌曲風格相似
14	風格較相似	風格、節奏較相似	風格、節奏較相似	風格較相似	風格較相似
15	曲風、節奏	語言、節奏	柔和感、節奏	風格較近似	速度、風格

表 4.6(b) (a)中各類型原因數量整理

	歌曲 1	歌曲 2	歌曲 3	歌曲 4	歌曲 5
節奏相關原因	6	2	3	1	10
音色相關原因	1	7	10	1	1
旋律相關原因	1	3	1	5	1
風格相關原因	7	5	2	6	5

從何者較為相似的選項中，可以看到選擇 musly 的結果較為相似的比較多，而選擇 onset RLCS 的較少，整體來說 musly 選出的歌曲更能讓人感覺相似，但是 onset RLCS 也在部分歌曲有所表現。比對選擇該首歌曲較相似的原因，可以發現選擇 musly 的傾向與音色相關(人聲的音色或樂器的音色)的原因，而選擇 onset RLCS 的傾向節奏相關(節奏或者速度)的原因，符合兩種系統所抽取的特徵值特色。

其中選擇 onset RLCS 較為相似的集中於第 1、5 首基準歌曲，這兩首基準歌曲與受測者所選出的最相似歌曲，都是有明確的鼓點打出節奏的歌曲，而 musly 在這兩首所選出的歌曲雖然音色相近，但是節奏、旋律相去甚遠而很少人選。

從整體結果來看，音色相似與節奏相似，何者更讓人實際聽到的感覺相似，對於不同歌曲可能不盡相同，因此以起音點為基礎的歌曲推薦方式，是能夠提供與以音色為基礎的方式不同的選擇的。

第五章 延伸研究與未來展望

本章中將簡單介紹除了前幾章所提到的東西之外，我們還做了哪些不同面向的嘗試，以及未來可以研究的相關目標。

5.1 延伸研究

本研究所使用的音樂分析方式，都是以起音點為基礎，因此除了以起音點來做歌曲相似度比較之外，我們還嘗試了是否能分析一首歌曲的起音點分布，來分類出不同的曲風。

我們找了一些有相同曲風的歌曲，嘗試畫出不同歌曲起音點間距的統計長條圖，希望能在相同曲風歌曲的起音點間距分布找到相似性。我們分別找了各 9 首探戈與華爾茲的歌曲來進行測試，結果如圖 5.1、5.2。

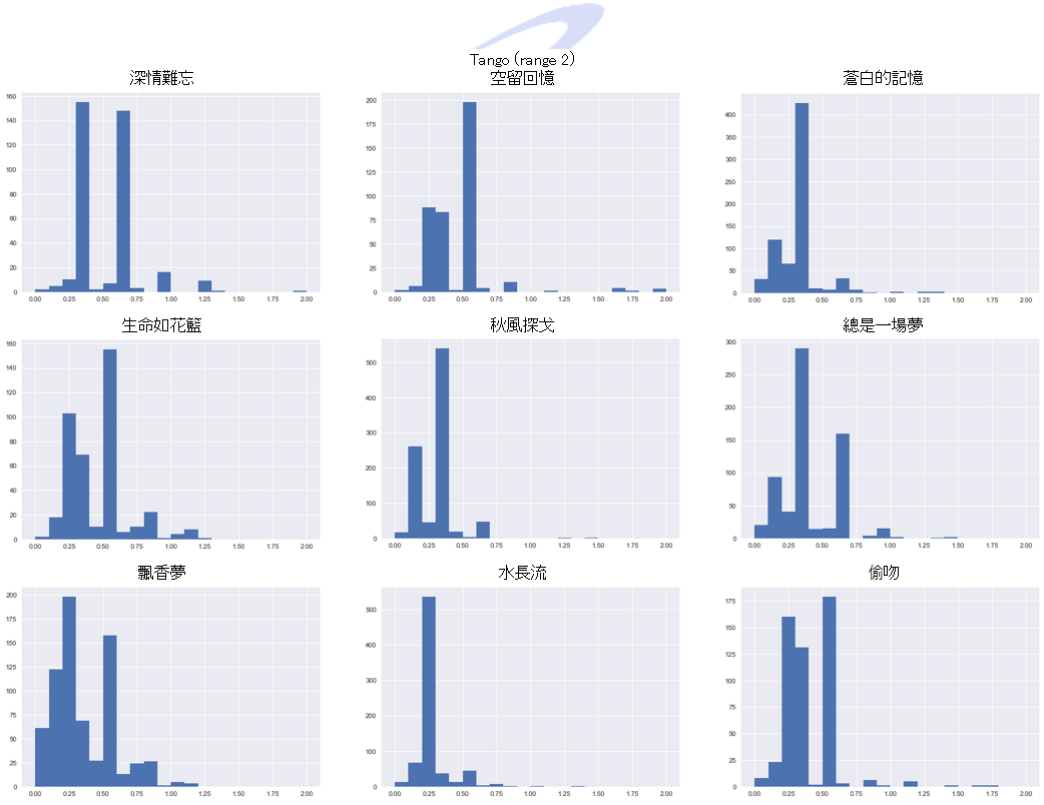

圖 5.1 探戈(Tango)曲風歌曲，起音點間距統計圖

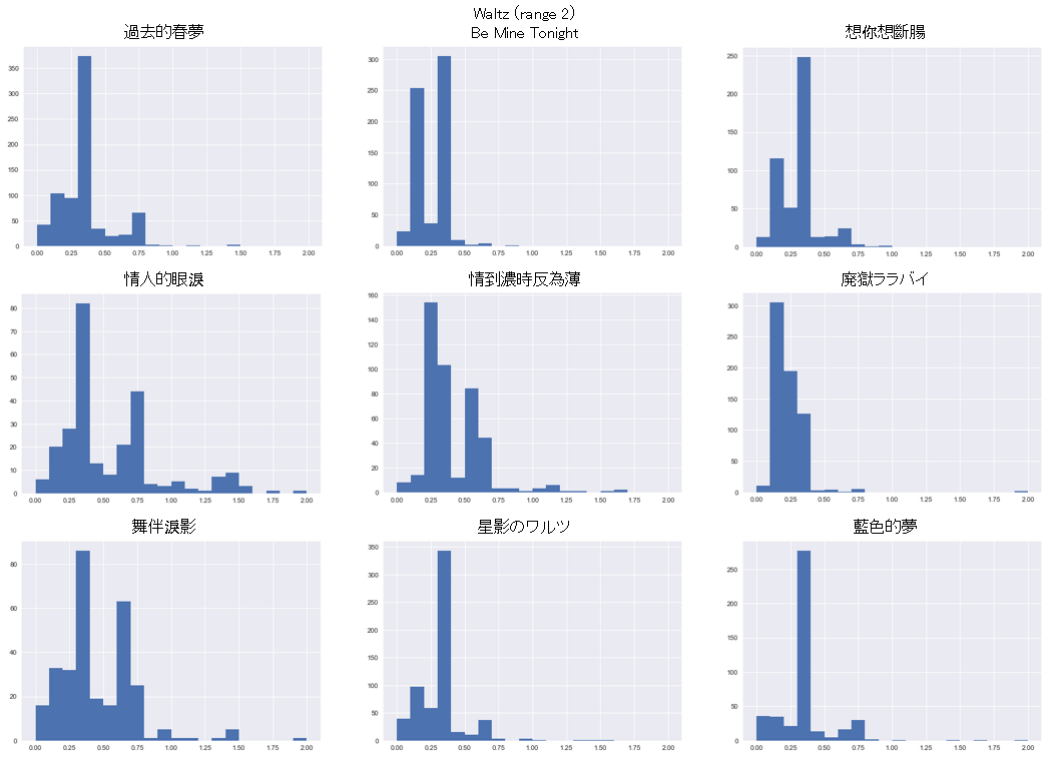

圖 5.2 華爾茲(Waltz)曲風歌曲，起音點間距統計圖

從測試結果來看，雖然有幾首歌統計出類似的分布，但是整體來看沒有明顯可以判斷為該曲風的特徵，單純統計起音點間距可能效果有限。原因可能在於純粹統計間距缺乏時間前後的連續關係，或者因為起音點偵測並不完全正確，當中存在許多雜訊，雜訊影響了時間間距的計算。

要利用起音點來判斷歌曲曲風，可能需要更多的方法來配合使用，這項測試只是我們想嘗試起音點的其他可能性，不是主要研究的目標，因此沒有繼續深入研究。

5.2 未來展望

以起音點做為歌曲相似度比對的方式，並以此建構歌曲推薦系統的可行性，以本研究所得到結果看來是可行的。儘管單純使用起音點來作為比對歌曲的特徵，得到的可能都是偏向節奏方面的相似性，但是如果與其他不同的方法相互配合，例如音色相似、旋律相似等，將不同方法建構為一套系統，依照不同的使用者需求，來提供在不同面向相似的歌曲，也能夠讓歌曲推薦系統的功能更為完善。

配合不同的歌曲比對方法來作歌曲推薦，是未來能嘗試的方向。又或者如前一節所述，將起音點偵測應用在其他分析歌曲的方面，例如以起點來判斷歌曲的曲風，或是利用起音點來找出一首歌中重複的段落，也都是可以繼續深入研究的目標。

參考文獻

[1] 趙若偉，利用起音點偵測搭配 RLCS 演算法進行歌曲相似度比對，國立台北科技大學碩士論文，民國 104 年。

[2] McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference, pp. 18-25. 2015.

[3] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In Proceedings of the 24th ACM International Conference on Multimedia, pp.1174-1178. 2016.

[4] Dominik Schnitzer. Musly is a fast and high-quality audio music similarity library written in C/C++. http://www.musly.org/

[5] J. P. Bello, et al, “A Tutorial on onset detection in music signals,” IEEE Trans. Speech and Audio Processing, vol. 13, no. 5, pp. 1035 – 1047, 2005.

[6] P. Masri and A. Bateman. "Improved modeling of attack transients in music analysis-resynthesis." Proceedings of the International Computer Music Conference, pp. 100 – 103, 1996.

[7] 黃信榮，哼唱式音樂搜尋系統中音符偵測與比對方法之研究，國立台灣科技大學資訊管理系研究所碩士論文，民國 99 年。

[8] J. Schlüter and S. Böck. Musical Onset Detection with Convolutional Neural Networks. In Proceedings of the 6th International Workshop on Machine Learning and Music, Prague, Czech Republic, 2013.

[9] CPJKU. Onset data set which can be used to tune/evaluate onset detection algorithms. https://github.com/CPJKU/onset_db

[10] H.-J. Lin, H.-H. Wu, C.-W. Wang, “Music matching based on rough longest common subsequence,” J. Info. Sci. Eng., vol. 27, no. 1, pp. 95 – 110, 2011.

[11] Cython C-Extensions for Python. http://cython.org/

[12] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In the proceedings of the 6th International Conference on Music Information Retrieval, ISMIR, 2005.

[13] Dominik Schnitzer, Arthur Flexer, Markus Schedl, Gerhard Widmer. Using mutual proximity to improve content-based audio similarity. In the proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR, 2011.